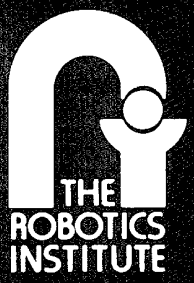
Cooperative Intelligent Software Agents

Dajun Zeng Katia Sycara

CMU-RI-TR-95-14



Carnegie Mellon University

The Robotics Institute

Technical Report

Cooperative Intelligent Software Agents

Dajun Zeng Katia Sycara

CMU-RI-TR-95-14

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

March, 1995

Accesion For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

†This research has been sponsored in part by ARPA Contract F33615-93-1-1330

Contents

List of Figures

List of Tables

Abstract

The availability of network-based information sources and services, along with the wide use of of World Wide Web, presents a great opportunity for enhanced problem solving support that incorporates information gathering into the problem solving framework. The concept of an intelligent information agent has been explored for information accessing and filtering tasks (e.g., filtering of newsgroup information). To provide support for more complex tasks that involve information gathering along with decision making capabilities, an agent must communicate and cooperate with other agents. One of the main issues is how to structure a multi-agent architecture that will allow access, filtering and fusing of information from many sources and services, integrated with flexible decision support. In this paper we present such an architecture. It has two layers: The top layer consists of *task-specific* software agents which help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents. The bottom layer consists of *information-specific* agents which provide intelligent access to a heterogeneous collection of information sources. The layered architecture has been implemented in the domain of everyday organizational tasks (e.g., hosting visitors, finding information about persons on the internet, managing personal calendars) at Carnegie Mellon. We illustrate this architecture and the agent coordination protocols in the hosting visitor domain.

1 Introduction

[1] Due to advances in networking technology, increasingly diverse and voluminous information is becoming available to decision makers. In effect, vast amounts of information in electronic form are currently freely available at a multitude of sites to anyone with access to the Internet. This presents the opportunity for enhanced levels of decision support that integrate information gathering and utilization into a decision support framework. Although this opportunity has been recognized by the decision support community [6], there has currently been limited investigation of problem solving frameworks where information is *actively* sought out in an open environment and is seamlessly integrated with problem solving and decision making [3, 9]. One of the main goals of our research in the PLEIADES group is to address this issue.

In most current information retrieval systems, users are assumed to know the information source and the form of their queries. Even though multiple data sources may be available, the user must decide which data source will most likely provide the correct answer, formulate the query according to the query language used by that data source and analyze the retrieved information. If the answer to a query is not adequate, then the user will either formulate another query or explore another data source. Moreover, when the retrieved data will be used to support decision making, the user must interpret and possibly reformat and enter the results into the decision support system. This mode of operation is very time consuming and inefficient, especially in the new information environment where: (1) The number and variety of data sources and services is dramatically increasing every day, (2) the information sources are multi-modal and distributed, and (3) the availability, type and reliability of information services are constantly changing. These characteristics of the new information environment necessitate new architectures for finding, retrieving, filtering and fusing information.

On the other hand, automated or semi-automated problem solving systems typically operate in a closed world environment, where almost every piece of information needed for problem solving is assumed to be known before problem solving starts or can be inferred by the system [2]. For ex-

[1]We want to thank Tom Mitchell, Rich Caruana, Dana Freitag, Matthew Glickman, Ken Lang, Sean Slittery, David Zabowski and other members of the PLEIADES project for interesting discussions. We also want to thank Gilad Amiri, Anandeep Pannu and Leonardo Garrido-Luna for doing much of the implementation.

ample, a planning system is assumed to know all the detailed specifications of available operators, initial state and goal state before it gets started. To realize the great opportunities for enhanced decision support presented by the availability of the new network-based electronic information environment, new problem solving architectures must be developed that integrate active information gathering with decision support functionality [3].

In this paper, we present a distributed framework where intelligent software agents cooperate asynchronously to perform goal-directed information retrieval and information integration in support of various tasks, such as finding information about people on the Internet, managing calendars and making arrangements to host visitors in an academic environment. We have implemented this architecture in the PLEIADES project. The broader goal of PLEIADES is to characterize and develop distributed agent-based architectures that are composed of *negotiating* and *learning* agents and apply them to provide logistics support for everyday tasks at Carnegie Mellon. Software agents coordinate and negotiate with each other to resolve disparities in the retrieved information. In addition, they learn from their users, the information sources, and each other. We believe that the proposed approach will help human users to harness the power of the information highway to support their decision making.

2 Distributed Architecture for Intelligent Information Retrieval and Problem Solving

We have developed a distributed architecture consisting of a collection of coordinating Intelligent Agents that coordinate with each other to actively seek, retrieve, filter, integrate and monitor information, and integrate the information into their problem solving to support a variety of users and tasks. The distributed architecture we have developed has been motivated by the following considerations:

- *distributed information sources:* Information sources available on-line are inherently distributed. Furthermore, these sources typically are of different modalities. Therefore it is natural to adopt a distributed architecture consisting of many software agents.

- *sharability:* Typically, user applications need to access several services or resources in an asynchronous manner in support of a variety of tasks. It would be wasteful to replicate agent information gathering or problem solving capabilities for each user and each application. It is desirable that the architecture support sharability of agent capabilities and retrieved information.

- *complexity hiding:* Often information retrieval in support of a task involves quite complex coordination of many different agents. To avoid overloading the user with a confusing array of different agents and agent interfaces, it is necessary to develop an architecture that hides the underlying distributed information gathering and problem solving complexity from the user.

- *modularity and reuseability:* Although software agents will be operating on behalf of their *individual patrons*—human users, pieces of agent code for a particular task can be copied from one agent to another and can be customized for their new user to take into consideration particular users' preferences or idiosyncrasies. One of the basic ideas behind the distributed agent-based approach is that software agents will be kept simple for ease of maintenance, initialization and customization. Another facet of reuseability is that pre-existing information services, whose implementation, query language and communication channels are beyond the control of user applications, could be easily incorporated in problem-solving.

- *flexibility:* software agents can interact in new configurations "on-demand", depending on the information requirements of a particular decision making task.

To address these issues, we have developed a distributed agent-based architecture that has two conceptual layers: The top layer consists of *task-specific* software agents which interact with the user to support user tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents. The bottom layer consists of *information-specific* software agents, which provides intelligent access to a heterogeneous collection of information resources. For example, a scheduling task-specific agent manages and updates a particular

user's appointment and meeting agenda. The general-purposed `finger` service module, which can extract useful information from the network `finger` utility given a user's login name and network address, can be viewed as an information-specific software agent.

A task-specific agent has the following knowledge: (1) model of the task domain, (2) procedural knowledge for performing the task, e.g., query decomposition, appointment scheduling capabilities, (3) knowledge about relevant task- or information-specific agents that it must coordinate with in support of its particular task, (4) protocols that enable coordination with the other relevant agents, and (5) strategies for conflict resolution and information fusion. If a task assistant is a *personal assistant* to a particular user, it possesses in addition, task-relevant preferences of its user. For details on automated acquisition of user preferences, see [1, 10]. Having the user interact only through a relevant task assistant hides the underlying distributed information gathering and problem solving complexity and frees the user from having to know of, access and interact with a plethora of information seeking agents in support of a task. For example, the hosting visitor task involves four information agents and four task agents. However, the user interacts directly only with the `Visitor Hoster` agent, the main task assistant for the visitor hosting task.

A typical information-specific agent knows: (1) model of the associated databases and meta-level information, such as size, average time it takes to answer a query and monetary cost of query processing, (2) procedures for accessing databases, (3) conflict resolution and information fusion strategies, and (4) protocols for coordination with the other relevant software agents. An information-specific agent also (1) caches answers to queries that are frequently asked, (2) determines how to manage the cached information, and (3) induces data base regularities and uses the learned summary knowledge during agent interactions. In addition, an information-specific agent can also be used as AI-enhanced gateway to externally available information services. For example, there are many pre-existing information services available online, such as various search engines for World Wide Web. To access these services, we have build information-specific agents that know how to to interact with the information sources and services and can inter-operate with our task-specific agents. In this way, the services that have been developed by others and are either pre-existing or newly added to the information environment can be utilized by PLEIADES to the full extent without affecting

any other part of the system. This capability supports scalability and maintenance of the system.

To answer queries submitted by other software agents or a human user, an information retrieval agent might need to decompose the query, find the relevant information resources, resolve any disparity among returned information and integrate the information. In case there are multiple information sources available, which might have replicated data and varying degrees of data correctness, processing time and costs, the information retrieval agent needs to choose the appropriate information sources relevant to an information request. If the chosen information sources fail to provide a useful answer, the information retrieval agent should seek and try other sources to re-do the data query. Because of these complexities of information retrieval, we view information retrieval as a planning task itself. In our distributed architecture, task-specific agents develop and monitor the execution of information retrieval plans as well as other task-related problem solving plans. A task-specific agent can decide when to actively seek new information about the environment or user, and in turn, utilize retrieved information for problem solving. This type of intelligent agents differs from traditional AI systems since information-seeking during problem solving is an inherently built-in part of the system. In effect, the planning and execution stages are interleaved since the retrieved information may change the planner's view of the outside world or alter the planner's inner belief system. In our system, information-specific agents are the information retrieval plan executors.

Due to space limitations, we briefly describe the distributed coordination processes in our multi-agent system. When a task-specific agent receives a task from a user or from another task-specific agent, it decomposes the task based on the domain knowledge it has and then delegates the subtasks to other task-specific agents or directly to information-specific agents. The task-specific agent will take responsibility of collecting retrieved data, resolving conflicts, coordinating among the related agents and finally reporting to the user. The agents who are responsible for assigned sub-tasks will either decompose these sub-tasks further based on their own domain knowledge or perform data retrieval (or possibly other domain-specific local problem solving activities). One of the most important features of this coordination paradigm is that information gathered from information retrieval steps will be incorporated in subsequent problem solving steps. Obviously, one of the major issues involved in multi-agent system is the problem of interoperabil-

ity and communication between the agents. In our framework, we use the KQML language [4] for inter-agent communication. In order to incorporate and utilize pre-existing software agents or information services that have been developed by others, we adopt the following strategy: If the agent is under our control, it will be built using KQML. If not, we build a gateway agent to handle different communication channels, different data and query formats, etc., to connect it with our agents. We are also implementing an advertisement mechanism and services registries that can be accessed by task-specific agents to help determine availability and location of desired information and services.

3 An Example Scenario: Organize a Visit

We illustrate our distributed system architecture and the interactions of the task-specific and information-specific agents in the scenario of hosting a visitor at CMU. Hosting a visitor involves arranging the visitor's schedule with faculty whose research interests match the interests that the visitor has expressed in his/her visit request. A different variation of the hosting visitor task has also been explored by Kautz and his colleagues at Bell Labs [5].

A visitor hosting agent should have the following capabilities: (1) It should automate information retrievals in terms of finding personnel information of potential appropriate meeting attendees. It should be able to access various on-line public databases and information resources at the disposal of the visit organizer. The system should also integrate the results obtained from various databases, clarify ambiguities (e.g., the same entity can be referred by different names in different partially replicated data bases) and resolve the conflicts which might arise from inconsistency between information resources. (2) It should create and manage the visitor's schedule as well as the meeting locations for the various appointments with the faculty members (e.g., a faculty's office, a seminar room). (3) It should possess a graphical user interface which can interact with the users. The GUI facilitates getting input from the user, presenting acquired information, asking for user confirmation as well as advising the user of the state of the system and its progress.

Our prototype system, called `Visitor Hoster`, supports the visitor hosting task. It takes as input a visit request, the tentative requested days for

the meeting and the research interests of the visitor. Its final output is a detailed schedule for the visitor of meetings with faculty members whose interests match the ones expressed in the visitor's request and who have been contacted and have agreed to meet with the visitor at times convenient for them. The scheduling task where multiple agents are involved is very complex and will not be described here. (For details, see [7].) To support the hosting visitors task, `Visitor Hoster` retrieves information from various heterogeneous information resources at $\mathcal{CMU}$ and also internet-based resources, such as remotely accessing plan files at sites external to $\mathcal{CMU}$ to extract information about people. The currently implemented information specific agents that are utilized in support of hosting visitors include: (1) `Finger` agent, which heuristically parses the retrieved information from remotely residing `finger` data bases. The *finger* agent is used to find information about the visitor so faculty can decide whether they are interested in meeting with him/her. The possible types of information that can be acquired in this way include: work title, research interests, work and home phone numbers, vacation plan, etc. (2) `Who's-Who` agent, which accesses on-line $\mathcal{CMU}$ who's who database through http-based queries to find information about potential faculty who should be contacted to see if they are interested in meeting with the visitor. (3) `Faculty Interests` agent, which can be used to retrieve information about the faculty members in the School of Computer Science at $\mathcal{CMU}$ with respect to their research interests. (4) `Computer-Science-Directory` agent, which can get the information about phone number, home address, etc. for all the members of the School of Computer Science at $\mathcal{CMU}$, including faculty members, staff and students. This information is utilized when available free slots in the visitor's calendar could be filled with meetings with students and staff in the requested research area.

The task-specific agents utilized in the hosting visitor task are: (1) `Visitor Hoster` agent, which accepts input from the user specifying the identity of the visitor, the visitor's area of interest, tentative visit date and visit duration. (2) `Scheduling` agent, which maintains a visitor's meeting schedule. (3) `Personnel Finder` agent, which accesses various information sources to find more specific information about the visitor and the faculty members that might be interested in meeting with the visitor. (4) `Interface` agent, which takes care of presenting acquired information from task or information specific agents to human users graphically.

We briefly present a visitor hosting scenario to illustrate the interactions

of the various agents. Suppose Gio Wiederhold wants to visit $\mathcal{CMU}$ CS department. Gio has requested that he would prefer to meet with $\mathcal{CMU}$ faculty interested in data bases. Relevant information about Gio, such as first and last name, affiliated organization, date and duration of his visit and his preference as to the interests of faculty he wants to meet with are input into the `Visitor Hoster` agent. Then the `Visitor Hoster` agent communicates with the `Faculty Interests` information specific agent to get a list of potential meeting candidates with research interests in databases. Based on the list of names returned in answer to this information gathering query, the `Personnel Finder` tries to collect more information for these potential meeting candidates so that they can be contacted and asked about whether they would be interested in meeting with Gio. For each potential meeting attendee, the `Personal Finder` agent spawns multiple queries to various information specific agents, i.e., the `Finger` agent, the `Who's-Who` agent and the `Computer-Science-Directory` agent simultaneously. These agents in turn translate the queries to match the format of the corresponding information resource. In particular, information is gathered about the faculty rank, office location, telephone and e-mail address of each of the potential meeting attendees. The `Personnel Finder` agent receives the replies for each potential attendee, merges the information and resolves any conflicts. It then sends the information to the `scheduling` agent. The `scheduling` agent selects the e-mail addresses of the most senior faculty [2] in Gio's area of interest (databases) and automatically sends them e-mail asking if they would like to meet with Gio on the date of his visit. If faculty members have personalized calendar management assistants (e.g., Tom Mitchell's Calendar Apprentice CAP [1, 8]), the `scheduling` agent communicates with those assistants [3]. For each contacted faculty member, his/her calendar assistant sends the `scheduling` agent information as to whether the faculty member agrees to meet with Gio and the preferred meeting location and time. Then the `scheduling` agent constructs a feasible schedule for Gio further coordinating with the meeting candidates' calendar assistants to resolve potential scheduling conflicts.

[2] Currently, the default duration of a meeting is one hour, so for a full day's visit, 8 faculty members are initially selected.

[3] For those faculty members who do not have a software calendar manager, the e-mail is in human readable form.

4 Conclusions

In this paper, we have presented an implemented distributed agent-based architecture to support decision making tasks that necessitate information retrieval from distributed net-based sources and utilization of the retrieved information into the decision support framework. Our approach has been implemented in the PLEIADES system that supports everyday organizational logistics tasks at $\mathcal{CMU}$. PLEIADES currently has 10 agents that coordinate with each other and with the user in cooperative information gathering and decision support. We believe that such flexible distributed architecture will be able to answer many of the challenges that face users as a result of the new, vast net-based information environment, such as locating, accessing, filtering and integrating information from disparate information sources and incorporating retrieved information into decision support tasks.

We are currently extending the PLEIADES capabilities to include (1) learning conflict resolution and negotiation strategies and (2) learning capabilities and reliability of inter-operating agents.

References

[1] Lisa Dent, Jesus Boticario, John McDermott, Tom Mitchell, and David Zabowski. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence.* AAAI, 1992.

[2] Oren Etzioni, Keith Golden, and Daniel Weld. Tractable closed world reasoning with updates. In *Proceedings of 4th International Conference on Principles of Knowledge Representation and Reasoning*, 1994.

[3] Oren Etzioni and Daniel Weld. A softbot-based interface to the internet. *Communications of the ACM*, 37(7), July 1994.

[4] Tim Finin, Rich Fritzson, and Don McKay. A language and protocol to support intelligent agent interoperability. In *Proceedings of the CE and CALS Washington 92 Conference*, June 1992.

[5] Henry A. Kautz, Bart Selman, and Michael Coen. Bottom-up design of software agents. *Communications of the ACM*, 37(7), July 1994.

[6] Jay Liebowitz. *Expert SYSTEMS for business and management.* Yourdon Press, Englewood Cliffs, N.J., 1990.

[7] JyiShane Liu and Katia Sycara. Distributed meeting scheduling. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, Georgia, August 13-16 1994.

[8] Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7), July 1994.

[9] Tim Oates, M. V. Nagendra Prasad, and Victor R. Lesser. Cooperative information gathering: A distributed problem solving approach. Technical Report UMass Computer Science Techincal Report 94-66, Depart of Computer Science, University of Massachusetts, 1994.

[10] Katia Sycara and Kazuo Miyashita. Case-based acquisition of user preferences for solution improvement in ill-structured domains. In *Proceedings of AAAI-94*, Seattle, Washington, August 1994. AAAI.